润乾
RAQSOFT

高性能计算数据库

润乾软件出品

www.raqsoft.com.cn

目录 CONTENTS

目录 CONTENTS

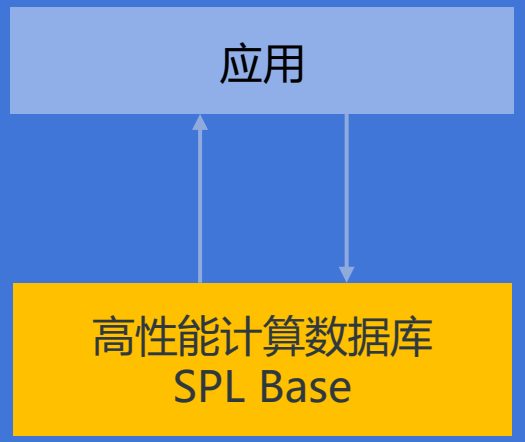
01

集算器是什么

数据计算中间件

集算器

- 一款专用的高性能计算数据库，简称：SPL Base
- SPL（Structured Process Language）是集算器内置的程序语言
- 高性能是集算器的主要特征



高性能计算数据库

集算器解决什么？

跑不完

半夜跑批跑不完，出错了来不及再来

月末年头更是担惊受怕...

查询慢

看个报表等10分钟，业务人员拍桌子...

人多了、时间跨度大了，就查不了了

反应钝

关联统计运算慢，界面拖拽迟钝；预汇总方案占用空间太大且功能盲区多

代价高

花了很多钱上了内存数据库，结果性能还是不理想...

无问单机与集群、无问国际大牌与国产新秀、无问MPP与HADOOP

集算器平均有数倍到数十倍的性能提升！

目录 CONTENTS

为什么
现有技术跑不快

计算性能的决定性因素

- 计算效率取决于硬件和软件两方面
- 软件性能就是算法效率
- 算法效率由算法设计和算法实现共同决定
- 好算法只想得出，实现不了也是徒劳
- 缺乏高性能机制的编程语言会限制好算法的实现

根本原因

没有火药，无论如何也造不出枪炮

结构化数据是重点

- 当前数据计算仍以业务系统产生的结构化数据为主
- 业界在提升结构化数据计算性能时主要依靠大内存、大集群
- 大内存、大集群的实质是纵向或横向提升硬件能力
- 而软件核心仍然使用SQL为主的关系代数体系
- SQL过于粗线条，很多高性能算法都无法实现

重点问题

SQL很难实现高性能结构化数据计算

SQL的问题

- SQL难以实现高效算法的原因在于其理论体系（关系代数）
- 理论上的缺陷很难通过工程实现来弥补

【举例】1亿行数据取前10名在SQL下会怎么做?

- SQL语句的原理是将所有数据大排序，然后取前10名，效率很低
- 大家都知道有不必大排序的办法实现这个运算，却无法用SQL表达
- 只能用指望数据库引擎自动优化，但复杂情况时数据库并不会优化

这就是为什么说，好算法不光要想出来，还要能实现

关于JAVA

相对SQL，JAVA可以实现高效算法，但JAVA实施计算的复杂度过高，可行性较低

```java
public static void groupCount(){
    groupBy mainJava = new groupBy();
    List<MainBean> list = mainJava.initList();
    List<MainBean> mainList = new ArrayList<MainBean>();
    Map<String, MainBean> map = new HashMap<String, MainBean>();
    MainBean totalBean = null;
    String employeeId = null;
    for(MainBean mainBean : list) {
        employeeId = mainBean.getEmployeeId();
        if (!map.containsKey(employeeId)) {
            totalBean = new MainBean();
            totalBean.setEmployeeId(employeeId);
            totalBean.setBuyNum(mainBean.getBuyNum());
            totalBean.setGoodsPrice(mainBean.getGoodsPrice());
            totalBean.setTotalBuyNum(mainBean.getTotalBuyNum());
            totalBean.setFreight(mainBean.getFreight());
            totalBean.setTotalGoodsPrice(mainBean.getGoodsPrice() * mainBean.getBuyNum());
            totalBean.setTotalPrice(mainBean.getGoodsPrice() * mainBean.getBuyNum() + mainBean.getFreight());
            mainList.add(totalBean);
            map.put(employeeId, totalBean);
        } else {
            totalBean = map.get(employeeId);
            totalBean.setTotalBuyNum(totalBean.getTotalBuyNum() + mainBean.getTotalBuyNum());
            totalBean.setFreight(totalBean.getFreight() + mainBean.getFreight());
            totalBean.setTotalGoodsPrice(totalBean.getTotalGoodsPrice() + mainBean.getGoodsPrice() * mainBean.getBuyNum());
            totalBean.setTotalPrice(totalBean.getTotalPrice() +
                    mainBean.getGoodsPrice() * mainBean.getBuyNum() + mainBean.getFreight());
        }
    }
```

这么长的代码仅仅实现了单个字段分组单个字段汇总的功能

高性能的本质

高性能算法想得出，写不出（SQL）

能写出又太难（JAVA）

高性能的本质是**开发效率**问题

要能想得到、方便写得出

目录 CONTENTS

集算器
为什么跑得快

集算器高性能来自于创新计算体系

【类比】计算1+2+3+...+100=?

普通人这么算

1+2=3
3+3=6
6+4=10
10+5=15
15+6=21
21+7=28
...

高斯这么算

1+100=101
2+99=101
...

一共有50个101
50*101=5050

高斯很聪明，想到了高效的算法，但更关键的是：那时人们已经发明了**乘法！**

前面的题：1亿行数据取前10名在SQL下会怎么做?

关系数据库的SQL就象只有加法的算术体系，而集算器的SPL则发明了乘法！

集算器还有更多乘法（高性能计算和存储库），人人都能成为高斯（快速实现高性能算法）

集算器的高性能来源于创新的代数体系

离散数据集模型

也就是集算器发明的“乘法”

集合化　离散性　深度集合化　有序性

解决难写的问题，从而实现高性能算法

敏捷语法更简单

计算目标：某支股票最长连续涨了多少交易日

```
select max(continuousDays)-1
from (select count(*) continuousDays
        from (select sum(changeSign) over(order by tradeDate) unRiseDays
                from (select tradeDate,
                        case when closePrice>lag(closePrice) over(order by tradeDate)
                        then 0 else 1 end changeSign
                      from stock) )
        group by unRiseDays)
```

SQL解法

- SQL在使用窗口函数的情况下嵌套三层完成；
- 读懂了吗？

	A
1	=stock.sort(tradeDate)
2	=0
3	=A1.max(A2=if(closePrice>closePrice[-1],A2+1,0))

SPL解法

其实这个计算很简单，按照自然思维：先按交易日排序（行1），然后比较当天收盘价比前一天高就+1，否则就清零，最后求个最大值（行3）

高性能算法与存储方案

内存查找

- 二分法
- 序号定位
- 位置索引
- 哈希索引
- 多层序号定位

外存数据集

- 文本文件的分段
- 集文件及倍增分段
- 数据类型
- 组表与列存
- 有序与补文件
- 数据更新及复组表

外存查找

- 二分法
- 哈希索引
- 排序索引
- 行存和带值索引
- 索引预加载
- 批量查找
- 返回集合的查找
- 多索引归并
- 全文检索

遍历技术

- 游标过滤
- 遍历复用
- 并行遍历
- 数据库并行加载
- 多路游标
- 分组汇总
- 聚合理解
- 冗余分组键

这里许多算法都是集算器的独创发明！

高性能算法与存储方案

有序遍历	外键关联	归并与连接	多维分析	集群
• 有序分组汇总	• 外键地址化	• 有序归并	• 部分预汇总	• 计算与数据分布
• 有序分组子集	• 临时地址化	• 分段归并	• 时间段预汇总	• 集群复组表
• 程序游标	• 外键序号化	• 关联定位	• 冗余排序	• 复写维表
• 前半序分组	• 内连接语法	• 附表	• 对位序列	• 分段维表
• 后半序分组	• 索引复用		• 标签位维度	• 冗余式容错
• 序号分组与可控分段	• 对位序列		• 内存标签异动	• 备胎式容错
	• 大维表查找			• 多作业负载均衡
• 索引排序	• 单边分堆			

这里许多算法都是集算器的独创发明！

高性能算法举例

聚合理解

	A	
1	=file(“data.ctx”).create().cursor()	
2	=A1.groups(;top(10,amount))	金额在前10名的订单
3	=A1.groups(area;top(10,amount))	每个地区金额在前10名的订单

高复杂度的排序转换为低复杂度的聚合

遍历复用

	A	
1	=file(“order.ctx”).create().cursor()	准备遍历
2	=channel(A1).groups(product;count(1):N)	配置复用计算
3	=A1.groups(area;sum(amount):amount)	遍历，并获得分组结果
4	=A2.result()	取出复用运算的结果

一次遍历可返回多个结果集

高性能存储

高性能数据存储

私有数据存储格式，集文件、组表

文件系统存储形式

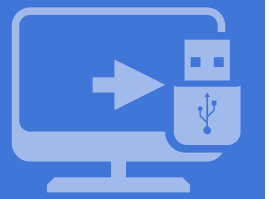
支持以树状目录方式按业务分类存储数据

集文件

倍增分段方式支持任意数量并行
自有高效压缩编码（减少空间；CPU占用少；安全）
泛型存储，允许集合数据

组表

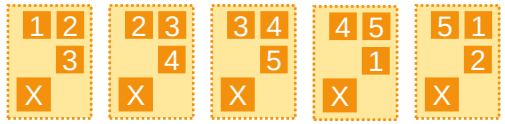
行列混合存储
有序存储提高压缩率和定位性能
高效智能索引
倍增分段方式支持任意数量并行
主子表合一减少存储与关联
排号键值实现高效定位关联



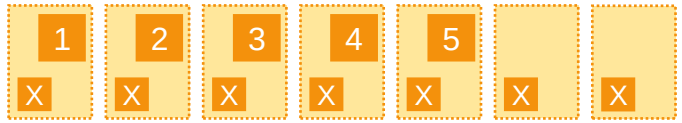
分布式计算

数据容错和计算容错

提供内外存两种数据容错机制，外存冗余式容错，内存备胎式容错



外存冗余式容错



内存备胎式容错

支持计算容错，节点故障时自动将该节点计算任务迁移掉其他节点继续完成

可控数据分布

用户可根据数据和计算任务的特点灵活定制数据分布及冗余方案，有效减少节点间数据传输量，以获得更高性能

无中心架构，避免单点失效

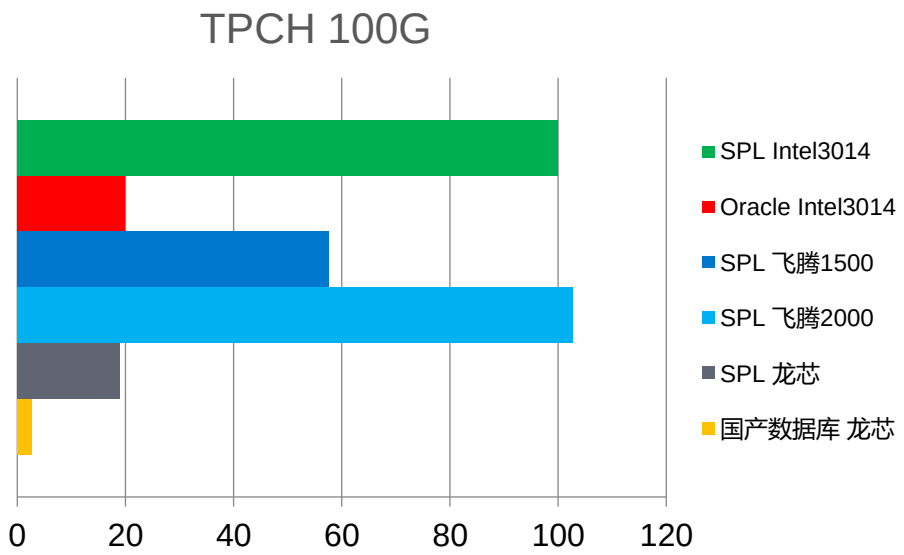
集群没有永久的中心主控节点，程序员用代码控制参与计算的节点

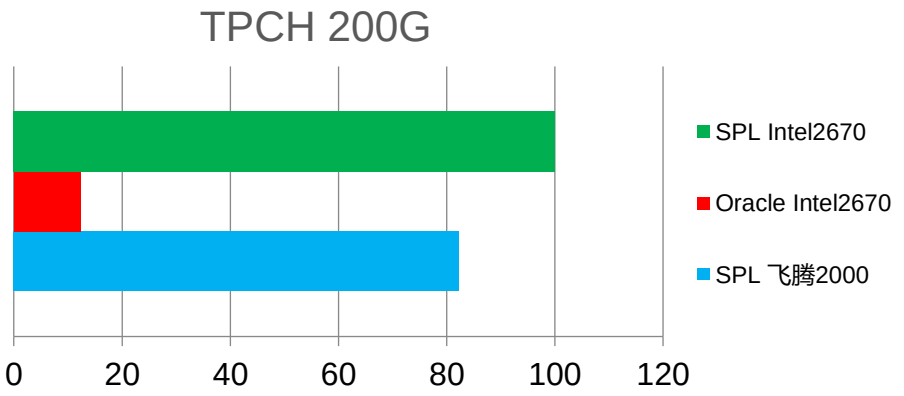
负载均衡能力

根据每个节点空闲程度（线程数量）决定是否分配任务，实现负担和资源的有效平衡

集算器性能表现

TPCH (单位：秒)	100G						200G		
	Intel 12核		飞腾16核	飞腾64核	龙芯8核		Intel 16核		飞腾64核
	SPL	Oracle	SPL	SPL	SPL	国产数据库	SPL	Oracle	SPL
1	38	131	62	19	275	507	40	325	36
2	4	27	8	6	18	247	8	73	13
3	16	222	33	22	97	4451	23	582	35
4	12	207	27	18	89	1790	21	454	43
5	20	225	36	24	72	1761	25	463	45
6	9	135	22	6	60	757	11	352	12
7	16	184	32	20	91	700	22	496	30
8	29	192	46	48	93	1611	29	485	80
9	68	234	125	65	517	1066	85	636	135
10	23	215	35	22	99	1634	34	493	42
11	5	33	12	6	29	165	9	63	11
12	25	184	72	38	173	647	52	464	55
13	57	37	114	85	335	2209	135	103	135
14	22	157	65	12	142	500	65	368	38
15	18	155	60	26	103	506	61	358	46
16	10	13	19	12	53	105	14	71	22
17	21	165	48	9	100	963	40	349	19
18	21	344	35	13	163	2382	25	966	26
19	23	154	65	12	137	518	60	345	29
20	18	175	57	11	110	594	55	442	20
21	233	326	222	190	901	3349	191	790	398
22	22	48	37	27	99	139	30	99	49
合计	710	3563	1232	691	3756	26601	1035	8777	1319





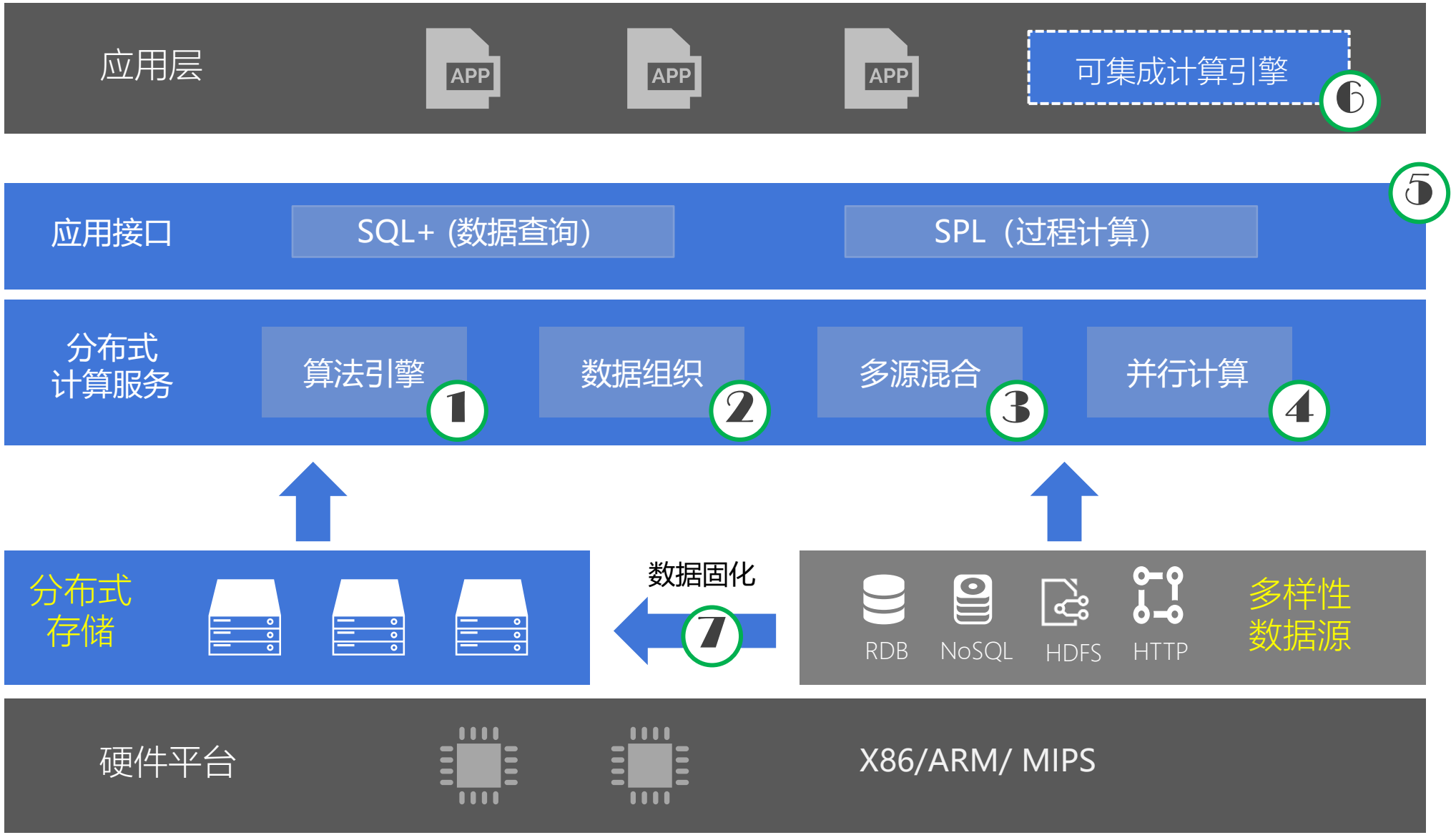
Intel3014 1.7G/12核/64G内存　　飞腾FT1500/16核/32G内存　　龙芯/8核/64G内存　　Intel2670 2.6G/16核/128G内存　　飞腾FT2000/64核/256G内存

目录 CONTENTS

04

应用架构与场景

应用架构


应用层
APP
APP
APP
可集成计算引擎
6
5
应用接口
SQL+（数据查询）
SPL（过程计算）
分布式
计算服务
算法引擎
1
数据组织
2
多源混合
3
并行计算
4
分布式
存储
数据固化
7
RDB
NoSQL
HDFS
HTTP
多样性
数据源
硬件平台
X86/ARM/ MIPS

应用场景

集算器（SPL Base）典型应用场景

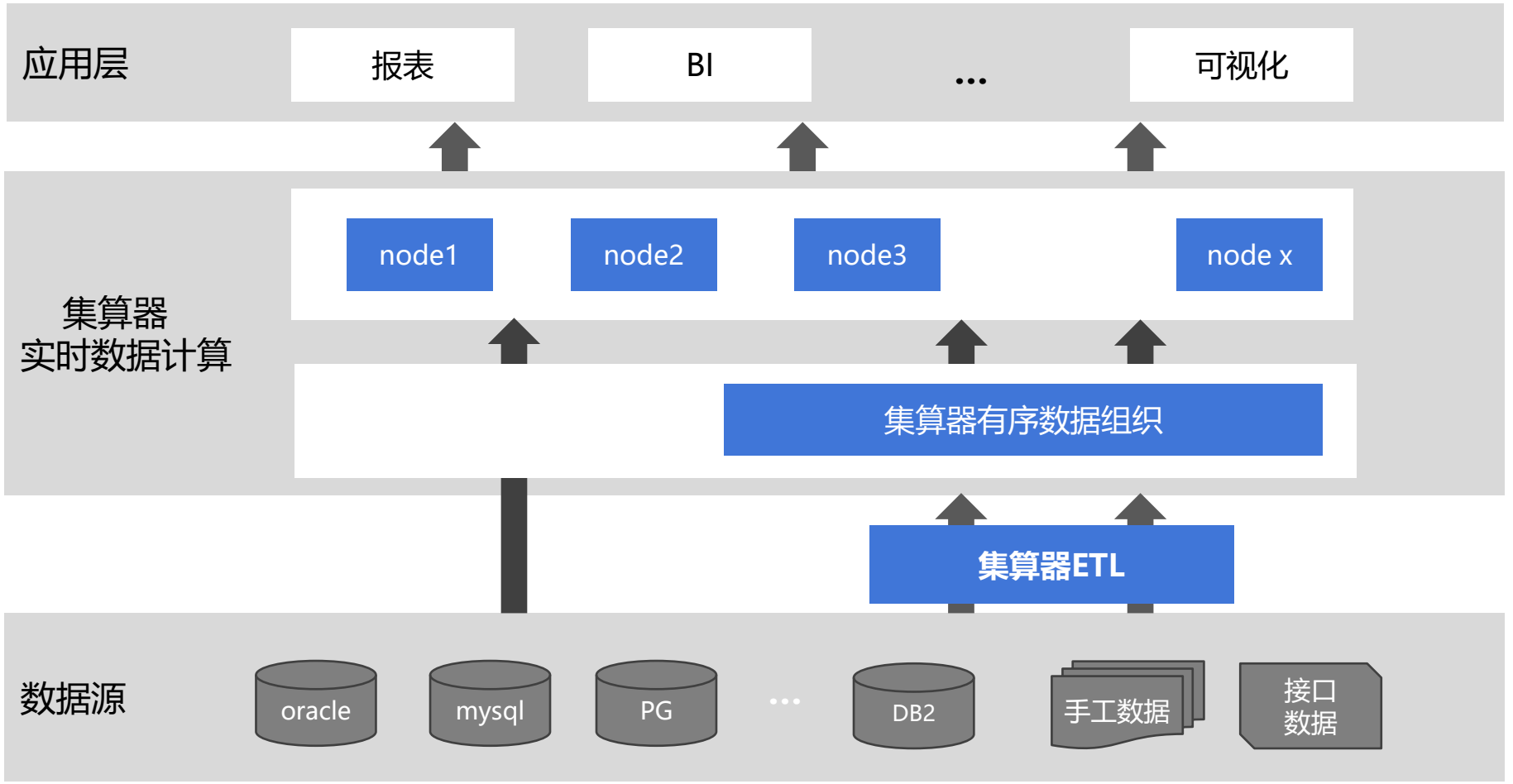
在线查询统计

交互分析取数

离线定时跑批

在线统计查询

【场景特征】多并发，业务可能复杂，秒级响应，大数据需集群支持

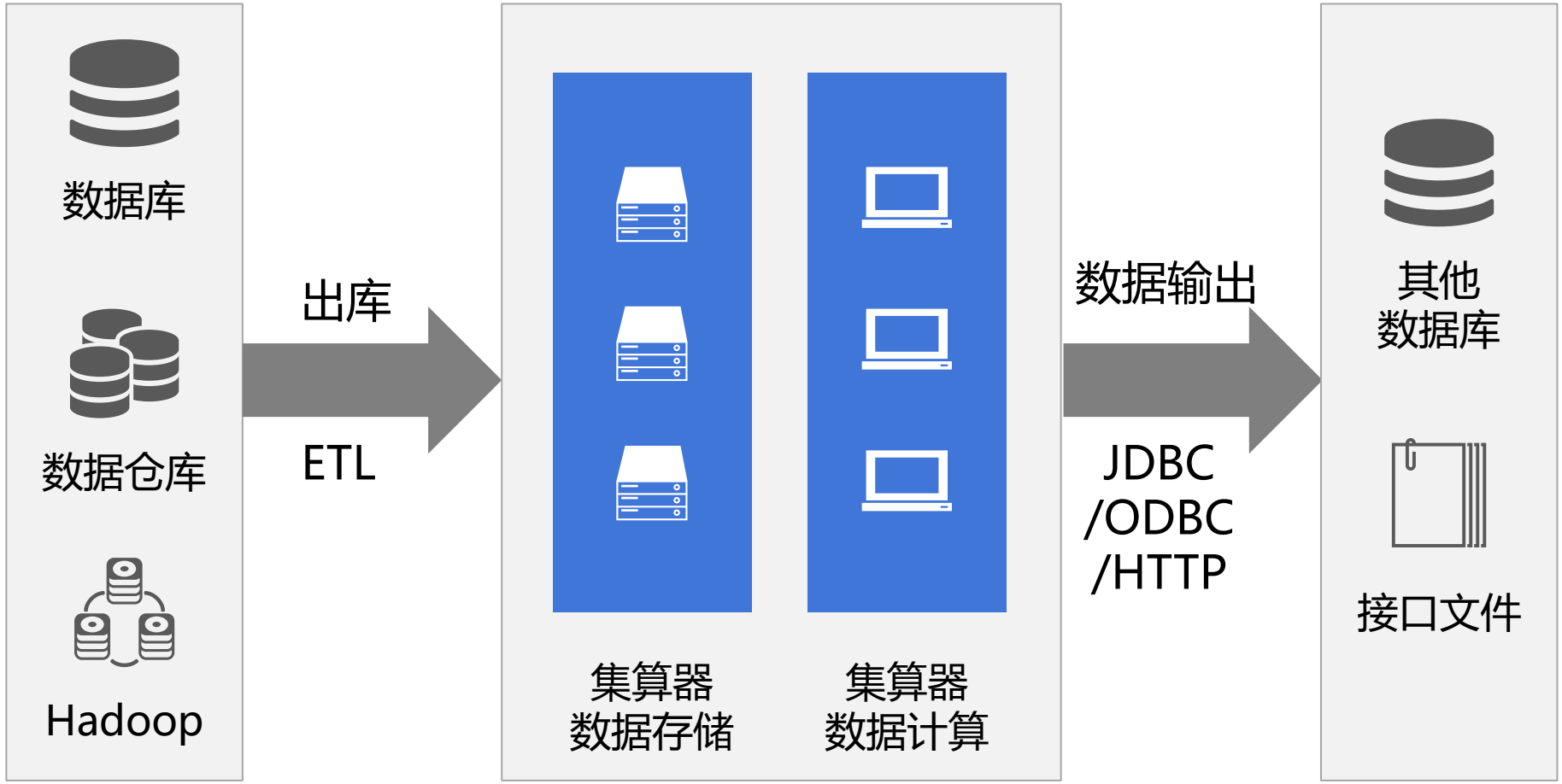


交互分析取数

【场景特征】无并发，实时性要求不高，会根据上一步的计算结果决定下一步计算

离线定时跑批

【场景特征】无并发，不必实时，数据量巨大，对时间窗口要求高



目录 CONTENTS

05

高性能常见问题

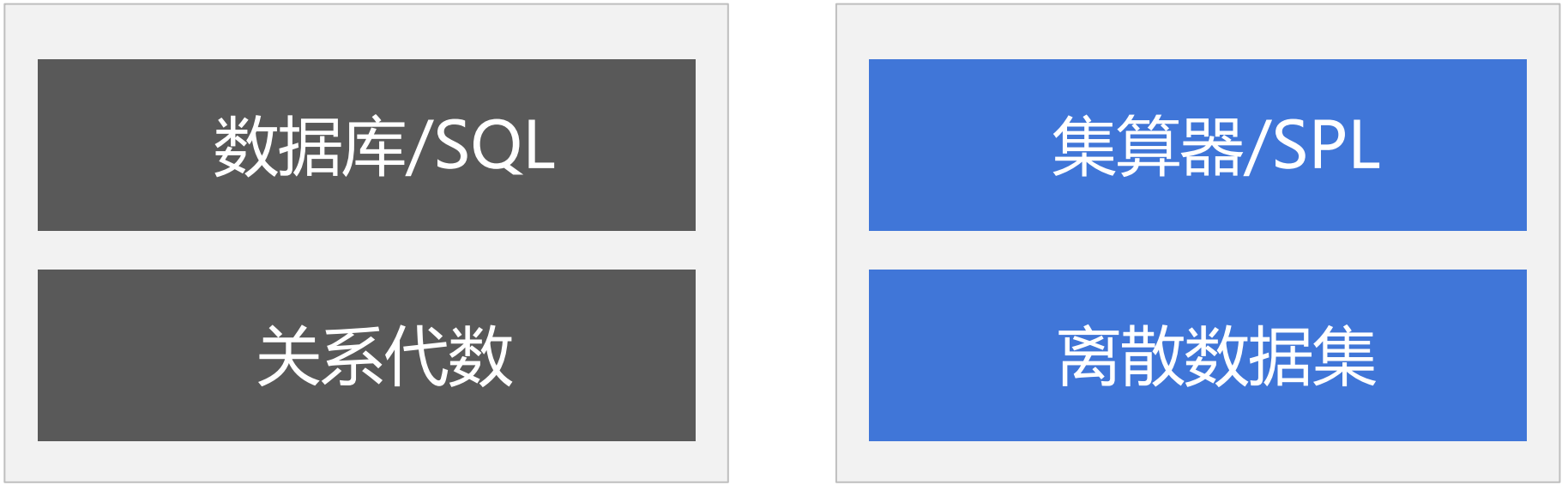
集算器要自行存储数据吗

必须！数据密集型计算的存储是性能保障，传统RDB和HADOOP的低效存储无法实现高性能

集算器针对内存、外存、集群都设计有专用高效数据组织方案，适应于多种运算场景

集算器基于开源或数据库技术?

集算器基于全新的计算模型，无开源技术可以引用，从理论到代码全部自主创新



基于创新理论的集算器不能再使用SQL实现高性能，SQL无法描述大部分低复杂度算法

仅对于运算形式规整的多维分析可提供高性能SQL接口，以适应各种前端BI工具

集算器的学习难度如何?

集算器专门用于性能优化，提供了专用的SPL语法

学习SPL不难，数小时即可掌握，数周就能熟练

难的是设计优化算法!

语法容易算法难

深度训练是关键

所以我们设计了

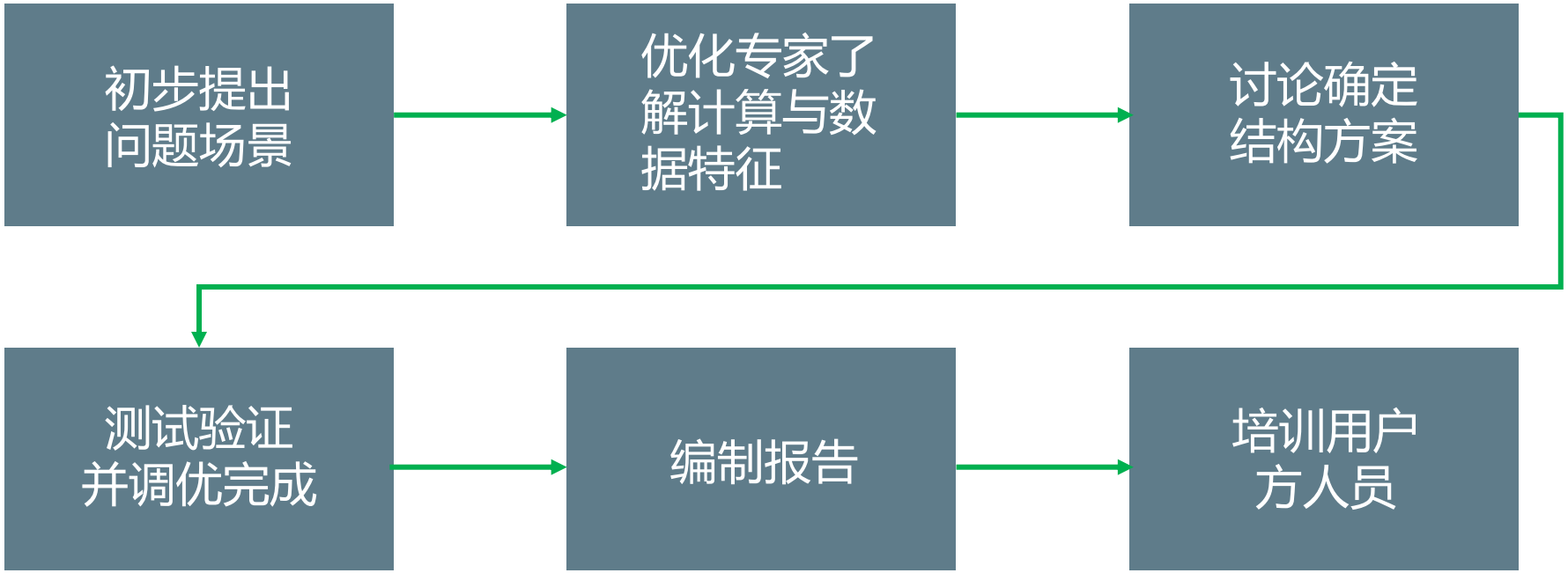
下面的优化流程

性能优化流程

最初的1-2个场景，由润乾高级工程师介入配合用户实现

大多数程序员习惯了SQL思维方式，不熟悉高性能算法，需要用一两个场景训练和理解

几十种性能优化套路经历过也就学会了，算法设计和实现并不是那么难



授人以鱼不如授人以渔！

更多资料

性能优化课程

http://www.raqsoft.com.cn/wx/course-performance-optimizing.html

应用案例

http://c.raqsoft.com.cn/article/1600822310565

润乾
RAQSOFT

THANKS

创新技术 推动应用进步

www.raqsoft.com.cn